METHOD AND APPARATUS FOR ACCESSING MULTIPLE VECTOR ELEMENTS IN PARALLEL

TECHNICAL FIELD

The present invention relates to a computer system comprising:

- a processor;
- a multi-port memory, the multi-port memory being accessible by the processor.

The present invention further relates to a method for transmitting a vector, in said computer system.

Still further, the present invention relates to a computer program for implementing said method.

BACKGROUND ART

Vector processing is a suitable technique for processing applications that have large computational demands. Vector processors provide high-level operations that work on vectors, i.e. linear arrays of numbers. Vector processors pipeline the operations on the individual elements of a vector. The pipeline includes not only the arithmetic operations, but also memory accesses and effective address calculations. In addition, most high-end vector processors allow multiple operations to be done at the same time, creating parallelism among the operations on different elements. Vector instructions have several important properties. First, the computations of each result are independent of the computations of previous results, allowing a very deep pipeline without generating any data hazards. Second, a vector instruction is equivalent to executing an entire loop, reducing the instruction bandwidth requirement. Third, the overhead of memory access is reduced, as in a single access a complete vector is retrieved instead of a data element. For these reasons, vector operations can be made faster than a sequence of scalar operations on the same number or data items. Typical applications where vector processing can be used is the domain of audio and video processing.

A vector memory system has a large datawidth, which allows retrieving a complete vector of data elements in one memory access using a single memory address. Subsequently, these data elements can be processed in parallel. However, several problems

can occur when retrieving data from a vector memory system. First, the problem of vector alignment is related to reading from a vector memory system data that cross vector boundaries. In that case the data can be retrieved by requesting the contents of two memory addresses, i.e. two vectors, and subsequently transfer the requested data to a new vector. Second, a problem arises when ordering of a set of elements of a vector, deviating from the order in which they are stored, is required. In case a vector is required with an ordered set of elements stored in different vectors, the contents of these vector have to be retrieved requiring at least two memory accesses followed by selection of the proper data elements. US 5,933,650 describes methods for alignment and ordering of vector elements. In the alignment of vector elements, one vector is loaded from a memory unit into a first register and another vector is loaded from the memory unit into a second register. A starting byte specifying the first byte of an aligned vector is determined. Next, a vector is extracted from the first register and the second register beginning from the first bit in the first byte of the first register continuing through the bits in the second register. Finally, the extracted vector is replicated into a third register such that the third register contains a plurality of elements aligned for vector processing. In the ordering of vector elements, a first vector is loaded from a memory unit into a first register and a second vector is loaded from the memory unit into a second register. Then, a subset of elements is selected from the first register and the second register. The elements from the subset are then replicated into the elements in the third register in a particular order suitable for subsequent vector processing.

It is a disadvantage of the prior art methods for alignment and ordering of vector elements that more than one read access to the vector memory system is required, increasing the overhead for obtaining vector data. Furthermore, additional hardware is required, e.g. for temporarily storing of vectors from which elements have to be selected for vector alignment or vector ordering.

DISCLOSURE OF INVENTION

An object of the invention is to provide an improved method for vector alignment and ordering of vector elements, resulting in a better performance of vector processors.

This object is achieved with a method for transmitting a vector, characterized in that the method comprises the steps of:

- passing a base memory address to an address configuration means;

- defining a set of memory addresses by the address configuration means using the base memory address and a configuration instruction for configuring the address configuration means;
- transmitting the vector to/from the multi-port memory using the set of memory addresses.

The method allows transmitting a complete vector to or from a multi-port memory, using a single base memory address. The data elements of a vector can be transmitted to or from arbitrary positions within the memory, improving flexibility and avoiding problems related to vector alignment and ordering of vector elements. Furthermore, the use of a multi-port memory in combination with said address configuration means reduces the instruction width. A complete vector can be transmitted using a single base memory address, whereas otherwise each memory address used by the multi-port memory should be present in the instruction. For certain types of processors, such as very large instruction word processors, reducing the code size is an important issue.

According to the invention a computer system is characterized in that the computer system further comprises an address configuration means, wherein the address configuration means is conceived to define a set of memory addresses using a base memory address and a configuration instruction for configuring the address configuration means, and wherein the multi-port memory is conceived to use the set of memory addresses. Complete vectors can be transmitted to or from the multi-port memory using one base memory address, which reduces memory overhead and increases the performance of the computer system.

Preferred embodiments of the invention are defined in the dependent claims. A computer program for implementing the method according to the invention for transmitting a vector is defined in Claim 8.

An embodiment of the computer system according to the invention is characterized in that:

- the address configuration means comprises: a plurality of register files arranged to be configured by the configuration instruction, and a plurality of address calculation units for calculating the set of memory addresses;
- the register files are accessible by the address calculation units;
- the address calculation units are coupled to the multi-port memory.

The configuration instruction configures the plurality of register files, and these register files can hold this configuration until the next configuration instruction is

executed. In between, this configuration can be used repeatedly, for example during execution of a loop of instructions.

An embodiment of the computer system according to the invention is characterized in that the configuration instruction comprises a set of offsets, each offset in combination with the base memory address defining a second memory address. The set of offsets can be directly loaded in the plurality of register files and used by the plurality of address calculation units, improving the performance of the address configuration means.

BRIEF DESCRIPTION OF THE DRAWINGS

The features of the described embodiments will be further elucidated and described with reference to the drawings:

Fig. 1 shows a schematic diagram of a computer system according to the invention.

Fig. 2 shows a schematic diagram of a memory system having a multi-port memory and an address configuration means.

DESCRIPTION OF PREFERRED EMBODIMENTS

Fig. 1 shows a block diagram of a computer system comprising a processor PROC, an address configuration unit ACU, a multi-port memory MEM and a system bus SB. The processor PROC, the address configuration unit ACU and the multi-port memory MEM are coupled via the system bus SB. During execution of instructions, the processor PROC may issue operations to access the multi-port memory MEM in order to read or write a vector with data elements. Prior to reading or writing a set of data elements from the multi-port memory MEM, the address configuration unit ACU should be configured by means of a configuration instruction, issued by the processor PROC. The configuration instruction configures the address configuration unit ACU so that it is capable of calculating a set of memory addresses specific for the set of data elements to be retrieved from the multi-port memory MEM, using a base memory address. The configuration of the address calculation unit ACU remains unchanged until a next configuration instruction is issued. After configuring the address configuration unit ACU, the processor issues a read operation, comprising a base memory address, and the latter is sent to the address calculation unit ACU. Subsequently the address calculation unit ACU calculates a set of memory addresses. These memory addresses are sent to the multi-port memory MEM via the system bus SB, followed by reading the data elements from the multi-port memory MEM. These data elements are

sent as a single vector to the processor PROC and used for further processing. In case the processor PROC issues a write operation, a base memory address is sent to the address configuration unit ACU. The address configuration unit ACU calculates a set of memory addresses, which are sent to the multi-port memory MEM, via the system bus SB. The data elements are also sent to the multi-port memory MEM via the system bus SB. In a next step, the data elements are written to the multi-port memory MEM. Prior to a next write or read operation it may be necessary to issue a new configuration instruction, depending on the set of memory addresses that is required. For example, in case a set of data elements must be read requiring the same set of memory addresses and applying the same base memory addresses, the configuration command does not have to be repeated. When a different base memory address is used, but the required configuration of the address configuration unit ACU remains identical, a new configuration instruction does not have to be issued either.

Fig. 2 shows a block diagram of a memory system MS, comprising a multi-port memory MEM and an embodiment of an address configuration unit ACU. The multi-port memory MEM comprises a RAM memory, four data input ports DatIn, four address ports Addr and four data output ports DatOut. The address configuration unit ACU comprises an address port AddrIn, four address calculation units AU, four register files RF and four data input ports DatIn. In this embodiment, the data inputs DatIn are shared data input ports for both the address configuration unit ACU and the multi-port memory MEM. The address input port AddrIn is coupled to the address calculation units AU, and the address calculation units AU are coupled to their corresponding address port Addr of the multi-port memory MEM. The data input ports DatIn are coupled to the register files RF. The register files RF are accessible by the address calculation units AU.

The multi-port memory MEM supports commands for reading and writing of data. Using the address ports Addr, data can be read from the RAM memory via the data output ports DatOut. The four data elements read from the data output ports DatOut can be combined into one vector. A set of four data elements can be written to the multi-port memory, via the data input ports DatIn and using the address ports Addr for memory addressing.

The address configuration units ACU support a configuration instruction, which specifies a set of offsets relative to a base memory address. When executing the configuration instruction, an offset value is written to each of the register files RF, via the corresponding data input port DatIn. Subsequently, the address calculation units AU fetch the offset value from their corresponding register file RF and store this value internally.

In case the processor PROC issues a read operation to the memory system MS, a base memory address is provided at the address port AddrIn. The address calculation units AU take the value of the base memory address from the address input port AddrIn and add their corresponding offset value. The address calculation units AU send the resulting set of memory addresses to the corresponding address ports Addr, and subsequently a read command is issued to the multi-port memory MEM. The resulting set of data elements is provided at the data output ports DatOut of the multi-port memory MEM. The processor PROC may also issue a write operation to the memory system MS in order to write a set of data elements to the RAM memory. The address port AddrIn receives a base memory address. The address calculation units AU calculate a set of memory addresses, using the base memory address and their corresponding offset value. The resulting set of memory addresses is sent to the corresponding address ports Addr of the multi-port memory MEM. The data elements are sent to the data input ports DatIn of the multi-port memory MEM. Subsequently, a write command is issued to the multi-port memory MEM and the data elements are written to the RAM memory.

In other embodiments, the configuration instruction may comprise a set of commands issued to the address configuration units AU for calculating a set of offsets.

Using a proper configuration instruction, the set of offsets received by the register files RF will be such that in combination with a base memory address the address calculation units AU are capable of defining an arbitrary set of memory addresses. Using this set of memory addresses, a set of data elements can be simultaneously written to or retrieved from the multi-port memory MEM. The memory system MS therefore behaves as a vector memory system, having the advantage of allowing retrieving a set of data elements from arbitrary memory locations using one base memory address. Furthermore, compared to a multi-port memory, the memory system MS has the advantage that using one memory address, a set of data elements can be addressed instead of requiring a set of memory addresses from an external source. As a result, the instruction width can be reduced, which is especially of interest for very large instruction word processors, where reduction of code size is an important issue.

It should be noted that the above-mentioned embodiments illustrate rather than limit the invention, and that those skilled in the art will be able to design many alternative embodiments without departing from the scope of the appended claims. In the claims, any reference signs placed between parentheses shall not be construed as limiting the claim. The word "comprising" does not exclude the presence of elements or steps other than those listed

in a claim. The word "a" or "an" preceding an element does not exclude the presence of a plurality of such elements. The invention can be implemented by means of hardware comprising several distinct elements, and by means of a suitably programmed computer. In the device claim enumerating several means, several of these means can be embodied by one and the same item of hardware. The mere fact that certain measures are recited in mutually different dependent claims does not indicate that a combination of these measures cannot be used to advantage.